UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/069,306	07/03/2002	Donald F. Hooper	10559-303US1	1999

20985 7590 11/03/2009
FISH & RICHARDSON, PC
P.O. BOX 1022
MINNEAPOLIS, MN 55440-1022

EXAMINER	
FENNEMA, ROBERT E	

ART UNIT	PAPER NUMBER
2183	

NOTIFICATION DATE	DELIVERY MODE
11/03/2009	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

PATDOCTC@fr.com

Office Action Summary	Application No.	Applicant(s)	
	10/069,306	HOOPER ET AL.	
	Examiner	Art Unit	
	Robert Fennema	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *15 July 2009*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,4-8,10-16,18-22,24 and 26* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1, 4-8, 10-16, 18-22, 24, and 26* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *7/15/09;10/1/09*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1, 4-8, 10-16, 18-22, 24, and 26 have been considered. Claims 1, 4-8, 10-13, 15, 18-22, and 24 amended as per Applicant's request.

2. Examiner notes the entry of the following papers:

- Amendment filed 7/15/2009
- IDS filed 7/15/2009
- IDS filed 10/1/2009

Claim Objections

3. In Claim 1, Line 3, Claim 15, Line 5, Claim 22, Line 3, and Claim 24, Line 6, instead of "a plurality of microengines, a microengine includes a context event arbiter", Examiner believes it would be more clear to claim "a plurality of microengines, each of the plurality of microengines including a context event arbiter", to make it more clear what is being claimed. Examiner has interpreted the claims as such based on the Applicant's arguments, however, the claims could also be interpreted such that only one microengine includes a context event arbiter, should the change not be effected.

4. In Claims 4-8, 10-11, 13, and 18-21 are objected to for "wakes up the swapped, current context". The current language is confusing, as the claim implies that a running context is waken. For example, instead of claiming "...which swaps out the current context corresponding to the first thread and wakes up the swapped, current context when an SRAM signal associated with the first thread is received" in Claim 4, it would

be much clearer to claim "which swaps out the first thread's context, and wakes up the first threads context when an SRAM signal associated with the first thread is received". The usage of the word "current", while used because of antecedent basis, is extremely confusing, because current is used as both an adjective and a label in the same claim, and Examiner believes that the claims would be much better if such language was eliminated or minimized to points where it will not be points of confusion.

5. In Claim 12, Line 3, "the thread" lacks antecedent basis.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1, 12, 14-16, 21, 24, and 26 are rejected under 35 U.S.C. 103(a) as being unpatentable over Borkenhagen et al. (USPN 6,212,544, herein Borkenhagen).

8. As per Claim 1, Borkenhagen teaches: A method of operating a multithreaded parallel processor comprising:

directing the processor having to swap based on a user-specified parameter specified in a context-swap instruction, a currently running context, corresponding to a first thread, in a specified microengine to let another context, corresponding to a

different thread that is ready to execute, execute in that microengine and cause a different context and associated program counter to be selected (Column 14, Lines 16-20, 29-33, and 41-44. Software (user) instructions can be executed to enable or disable specific events to cause a context switch, which causes a new context, which inherently has its own program counter, to be executed in place of the original), with the swapped first thread automatically re-enabled to run at some subsequent context arbitration point (Abstract, this is how context switching works),

wherein directing the processor comprises waking up the swapped out context when the user-specified parameter specified in the context-swap instruction is activated, with the user-specified parameter specifying an occurrence of an event (Column 14, Line 16 – Column 15, Line 15, each thread/context has a series of enable bits, and when the event defined by the enable bit is activated, the contexts switch (either sleeping or waking the context, depending on the situation)), the occurrence of the event having been communicated to the context event arbiter (Column 5, Lines 10-16, the thread switch logic is responsible for the threads switching, so must be aware of the conditions for thread switching), but fails to teach:

a plurality of microengines, a microengine including a context event arbiter.

While Borkenhagen teaches the functionality of Claim 1, Borkenhagen does not explicitly teach a microengine which contains a context event arbiter, as the Thread Switch Logic would correspond to the context event arbiter, but there is only one of those on the processor, thus, there are not a plurality of microengines each with an arbiter. However, in Column 2, Lines 27-30, Borkenhagen suggests that performance

can be increased by using multiple processors, shared on the same circuit (also known as a Multi-Core Processor). Borkenhagen discloses a means to further increase the performance of each of these individual processors, however, one of ordinary skill in the art would have been motivated to implement several of the processors described by Borkenhagen into a single, multi-core processor, as suggested by Borkenhagen himself, in order to further increase the system performance. In this case, the parallel processor corresponds to the multicore processor, and each microengine corresponds to the individual processor cores described by Borkenhagen, each with their own context arbiter to change contexts.

9. As per Claim 12, Borkenhagen teaches: The method of claim 1 wherein the user-specified parameter specifies "kill" which prevents the current context corresponding to the first thread or the first thread from executing again until an appropriate enable bit for the thread is set in a CTX_ENABLES register (Column 14, Lines 41-44, an instruction can disable all event conditions, preventing it from being switched back in).

10. As per Claim 14, Borkenhagen teaches: The method of claim 1 wherein directing further comprises:

in response to an optional_token "defer one" specified in the context-swap instruction, executing an additional instruction in an instruction stream of the currently running context before the context is swapped (Column 17, Lines 1-27, if certain

conditions are met, one instruction must be executed prior to any swap occurring).

11. As per Claim 15, Borkenhagen teaches: A method of operating a multithreaded parallel processor, the method comprising:

specifying an occurrence of an event with a user-specified parameter (Column 14, Lines 16-20, 29-33, and 41-44. Software (user) instructions can be executed to enable or disable specific events to cause a context switch), the occurrence of the event having been communicated to a context event arbiter corresponding to a microengine (Column 5, Lines 10-16, the thread switch logic is responsible for the threads switching, so must be aware of the conditions for thread switching),

receiving a user-specified parameter specified in a context-swap instruction (Column 14, Lines 16-20, 29-33, and 41-44. Software (user) instructions can be executed to enable or disable specific events to cause a context switch);

performing a swapping operation to cause an executing context process corresponding to a first thread to be swapped with a different context and associated program counter, corresponding to a different thread that is ready to execute (Column 5, Lines 10-12, different threads inherently have their own program counters), the swapped first thread being automatically re-enabled to run at some subsequent context arbitration point (Abstract, this is how context switching works); and

waking up the swapped out context when the user-specified parameter specified in the context-swap instruction is activated (Column 14, Line 16 – Column 15, Line 15, each thread/context has a series of enable bits, and when the event defined by the

enable bit is activated, the contexts switch (either sleeping or waking the context, depending on the situation)), but fails to teach:

the multithreaded parallel processor comprising a plurality of microengines, a microengine including a context event arbiter.

While Borkenhagen teaches the functionality of Claim 15, Borkenhagen does not explicitly teach a microengine which contains a context event arbiter, as the Thread Switch Logic would correspond to the context event arbiter, but there is only one of those on the processor, thus, there are not a plurality of microengines each with an arbiter. However, in Column 2, Lines 27-30, Borkenhagen suggests that performance can be increased by using multiple processors, shared on the same circuit (also known as a Multi-Core Processor). Borkenhagen discloses a means to further increase the performance of each of these individual processors, however, one of ordinary skill in the art would have been motivated to implement several of the processors described by Borkenhagen into a single, multi-core processor, as suggested by Borkenhagen himself, in order to further increase the system performance. In this case, the parallel processor corresponds to the multicore processor, and each microengine corresponds to the individual processor cores described by Borkenhagen, each with their own context arbiter to change contexts.

12. As per Claim 16, Borkenhagen teaches: The method of claim 15 wherein performing comprises swapping a currently running context in a specified microengine

to let another context execute in that microengine (Column 5, Lines 10-12).

13. As per Claim 21, Borkenhagen teaches: The method of claim 15 further comprising:

in response to an optional_token "defer one" specified in the context-swap instruction, executing an additional instruction in an instruction stream of the currently running context corresponding to the first thread before the currently-running context is swapped (Column 17, Lines 1-27, if certain conditions are met, one instruction must be executed prior to any swap occurring).

14. As per Claim 24, Borkenhagen teaches: A computer program product residing on a computer readable storage device for causing a multithreaded parallel processor to perform a function, the computer program product comprising instructions causing the processor to:

specify an occurrence of an event with a user-specified parameter (Column 14, Lines 16-20, 29-33, and 41-44. Software (user) instructions can be executed to enable or disable specific events to cause a context switch), the occurrence of the event having been communicated to a context event arbiter corresponding to a microengine (Column 5, Lines 10-16, the thread switch logic is responsible for the threads switching, so must be aware of the conditions for thread switching),

receive a user-specified parameter specified in a context-swap instruction (Column 14, Lines 16-20, 29-33, and 41-44. Software (user) instructions can be executed to enable or disable specific events to cause a context switch);

perform a swapping operation to cause an executing context process corresponding to a first thread to be swapped with a different context and associated program counter, corresponding to a different thread that is ready to execute (Column 5, Lines 10-12, different threads inherently have their own program counters), the swapped first thread is automatically re-enabled to run at some subsequent context arbitration point (Abstract, this is how context switching works); and

wake up the swapped out context when the voluntary swap parameter specified in the context-swap instruction is activated (Column 14, Line 16 – Column 15, Line 15, each thread/context has a series of enable bits, and when the event defined by the enable bit is activated, the contexts switch (either sleeping or waking the context, depending on the situation)), but fails to teach:

the multithreaded parallel processor comprising a plurality of microengines, a microengine including a context event arbiter.

While Borkenhagen teaches the functionality of Claim 24, Borkenhagen does not explicitly teach a microengine which contains a context event arbiter, as the Thread Switch Logic would correspond to the context event arbiter, but there is only one of those on the processor, thus, there are not a plurality of microengines, each with an arbiter. However, in Column 2, Lines 27-30, Borkenhagen suggests that performance can be increased by using multiple processors, shared on the same circuit (also known

as a Multi-Core Processor). Borkenhagen discloses a means to further increase the performance of each of these individual processors, however, one of ordinary skill in the art would have been motivated to implement several of the processors described by Borkenhagen into a single, multi-core processor, as suggested by Borkenhagen himself, in order to further increase the system performance. In this case, the parallel processor corresponds to the multicore processor, and each microengine corresponds to the individual processor cores described by Borkenhagen, each with their own context arbiter to change contexts.

15. As per Claim 26, Borkenhagen teaches: The method of claim 1 wherein the user-specified parameter specifies "voluntary" (Column 18, Lines 55-59).

16. Claims 4-8, 10-11, 13, 18-20, and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Borkenhagen, in view of Official Notice.

17. As per Claim 22, Borkenhagen teaches: A parallel processor that can execute multiple contexts and that comprises:

a program counter for each executing context (inherent);

an arithmetic logic unit coupled to the register stack (Figure 3, the execution units) and a program control store that stores a context swap instruction (Column 16, Lines 20-22) that causes the processor to:

receive a user-specified parameter in the context swap instruction specified in the context swap instruction (Column 14, Lines 16-20, 29-33, and 41-44. Software (user) instructions can be executed to enable or disable specific events to cause a context switch);

perform a swap operation to cause an executing context process corresponding to a first thread to be swapped with a different context and associated program counter, corresponding to a different thread that is ready to execute (Column 5, Lines 10-12, different threads inherently have their own program counters), the swapped first thread is automatically re-enabled to run at some subsequent context arbitration point (Abstract, this is how context switching works); and

wake up the swapped out context when the user-specified parameter specified in the context-swap instruction is activated, the user-specified parameter specifying an occurrence of an event (Column 14, Line 16 – Column 15, Line 15, each thread/context has a series of enable bits, and when the event defined by the enable bit is activated, the contexts switch (either sleeping or waking the context, depending on the situation)), the occurrence of the event having been communicated to the context event arbiter (Column 5, Lines 10-16, the thread switch logic is responsible for the threads switching, so must be aware of the conditions for thread switching), but fails to teach:

a plurality of microengines, a microengine including a context event arbiter; and

a register stack.

While Borkenhagen teaches the functionality of Claim 22, Borkenhagen does not explicitly teach a microengine which contains a context event arbiter, as the Thread

Switch Logic would correspond to the context event arbiter, but there is only one of those on the processor, thus, there are not a plurality of microengines, each with an arbiter. However, in Column 2, Lines 27-30, Borkenhagen suggests that performance can be increased by using multiple processors, shared on the same circuit (also known as a Multi-Core Processor). Borkenhagen discloses a means to further increase the performance of each of these individual processors, however, one of ordinary skill in the art would have been motivated to implement several of the processors described by Borkenhagen into a single, multi-core processor, as suggested by Borkenhagen himself, in order to further increase the system performance. In this case, the parallel processor corresponds to the multicore processor, and each microengine corresponds to the individual processor cores described by Borkenhagen, each with their own context arbiter to change contexts.

Borkenhagen further teaches a set of registers (Column 8, Line 1), but does not teach that these registers are a "stack". However, Examiner is taking Official Notice that a stack is one of the few common ways to arrange a register, that one of ordinary skill in the art would be aware of this, and thus, one of ordinary skill in the art would have been motivated to implement the registers as a stack, as implementing the registers as a stack instead of an array is a design choice, and not a patentable or innovative distinction.

18. As per Claim 4, Borkenhagen teaches: The method of claim 1, but fails to teach:

wherein the user-specified parameter specifies "sram Swap", and which swaps out the current context corresponding to the first thread and wakes up the swapped, current context when an SRAM signal associated with the first thread is received.

While Borkenhagen does not specifically teach that one of the parameters is a "sram swap", dealing with the SRAM signal, Borkenhagen does teach a large number of signals that can trigger a context switch, as shown in Columns 14 and 15, and also notes that there are a large number of extra bits available for future use. Examiner is taking Official Notice that one of ordinary skill in the art would have been motivated to make use of any number of other signals, including the SRAM signal, to trigger a context switch, if their particular implementation deemed it beneficial. Given that Borkenhagen provides both the motivation and the structure to add in additional signals, Examiner believes it would be an obvious distinction to use the SRAM signal to switch threads.

19. As per Claim 5, Borkenhagen teaches: The method of claim 1, but fails to teach:

wherein the user-specified parameter specifies "sdram Swap," which swaps out the current context corresponding to the first thread and wakes up the swapped, current context when an SDRAM signal associated with the first thread is received.

While Borkenhagen does not specifically teach that one of the parameters is a "sdram swap", dealing with the SDRAM signal, Borkenhagen does teach a large number of signals that can trigger a context switch, as shown in Columns 14 and 15, and also notes that there are a large number of extra bits available for future use.

Examiner is taking Official Notice that one of ordinary skill in the art would have been motivated to make use of any number of other signals, including the SDRAM signal, to trigger a context switch, if their particular implementation deemed it beneficial. Given that Borkenhagen provides both the motivation and the structure to add in additional signals, Examiner believes it would be an obvious distinction to use the SDRAM signal to switch threads.

20. As per Claim 6, Borkenhagen teaches: The method of claim 1, but fails to teach:

wherein the user-specified parameter specifies "FBI" which swaps out the current context corresponding to the first thread and wakes up the swapped, current context when an FBI signal associated with the first thread is received indicating that an FBI CSR, Scratchpad, TFIFO, or RFIFO operation has completed.

While Borkenhagen does not specifically teach that one of the parameters is “fbi”, dealing with the FBI signal, Borkenhagen does teach a large number of signals that can trigger a context switch, as shown in Columns 14 and 15, and also notes that there are a large number of extra bits available for future use. Examiner is taking Official Notice that one of ordinary skill in the art would have been motivated to make use of any number of other signals, including the FBI signal, to trigger a context switch, if their particular implementation deemed it beneficial. Given that Borkenhagen provides both the motivation and the structure to add in additional signals, Examiner believes it would be an obvious distinction to use the FBI signal to switch threads.

21. As per Claim 7, Borkenhagen teaches: The method of claim 1, but fails to teach:

wherein the user-specified_parameter specifies "seq_numl_change/seq_num2_change", which swaps out the current context corresponding to the first thread and wakes up the swapped, current context when a value of a sequence number changes.

While Borkenhagen does not specifically teach that one of the parameters is a "seq_numl_change/seq_num2_change", dealing with a sequence number, Borkenhagen does teach a large number of signals that can trigger a context switch, as shown in Columns 14 and 15, and also notes that there are a large number of extra bits available for future use. Examiner is taking Official Notice that one of ordinary skill in the art would have been motivated to make use of any number of other signals, including a sequence number, to trigger a context switch, if their particular implementation deemed it beneficial. Given that Borkenhagen provides both the motivation and the structure to add in additional signals, Examiner believes it would be an obvious distinction to use a sequence number to switch threads.

22. As per Claim 8, Borkenhagen teaches: The method of claim 1, but fails to teach:

wherein the user-specified parameter specifies "inter_thread" which swaps out the current context corresponding to the first thread and wakes up the swapped, current context when an interthread signal associated with the first thread is received.

While Borkenhagen does not specifically teach that one of the parameters is an "inter_thread" signal, dealing with the interthread signal, Borkenhagen does teach a

large number of signals that can trigger a context switch, as shown in Columns 14 and 15, and also notes that there are a large number of extra bits available for future use. Examiner is taking Official Notice that one of ordinary skill in the art would have been motivated to make use of any number of other signals, including the interthread signal, to trigger a context switch, if their particular implementation deemed it beneficial. Given that Borkenhagen provides both the motivation and the structure to add in additional signals, Examiner believes it would be an obvious distinction to use the interthread signal to switch threads.

23. As per Claim 10, Borkenhagen teaches: The method of claim 1, but fails to teach:

wherein the user-specified parameter specifies "auto_push" which swaps out the current context corresponding to the first thread and wakes up the swapped, current context when SRAM transfer read register data has been automatically pushed by a FBus interface.

While Borkenhagen does not specifically teach that one of the parameters is an "auto_push" signal, dealing with the FBus signal, Borkenhagen does teach a large number of signals that can trigger a context switch, as shown in Columns 14 and 15, and also notes that there are a large number of extra bits available for future use. Examiner is taking Official Notice that one of ordinary skill in the art would have been motivated to make use of any number of other signals, including the FBus signal, to trigger a context switch, if their particular implementation deemed it beneficial. Given that Borkenhagen provides both the motivation and the structure to add in additional

signals, Examiner believes it would be an obvious distinction to use the FBus signal to switch threads.

24. As per Claim 11, Borkenhagen teaches: The method of claim 1, but fails to teach:

wherein the user-specified parameter specifies "startreceive" which swaps out the current context corresponding to the first thread and wakes up the swapped, current context when new data in a receive FIFO is available for the first thread to process.

While Borkenhagen does not specifically teach that one of the parameters is a "startrecieve" signal, dealing with FIFO availability, Borkenhagen does teach a large number of signals that can trigger a context switch, as shown in Columns 14 and 15, and also notes that there are a large number of extra bits available for future use. Examiner is taking Official Notice that one of ordinary skill in the art would have been motivated to make use of any number of other signals, including whether FIFO data is available, to trigger a context switch, if their particular implementation deemed it beneficial. Given that Borkenhagen provides both the motivation and the structure to add in additional signals, Examiner believes it would be an obvious distinction to use a FIFO available signal to switch threads.

25. As per Claim 13, Borkenhagen teaches: The method of claim 1, but fails to teach:

wherein the user-specified parameter specifies "pci" which swaps out the current context corresponding to the first thread and wakes up the swapped, current context when a PCI unit signals that a DMA transfer has been completed.

While Borkenhagen does not specifically teach that one of the parameters is a "pci" signal, dealing with DMA transfers, Borkenhagen does teach a large number of signals that can trigger a context switch, as shown in Columns 14 and 15, and also notes that there are a large number of extra bits available for future use. Examiner is taking Official Notice that one of ordinary skill in the art would have been motivated to make use of any number of other signals, including whether the PCI signal has activated, to trigger a context switch, if their particular implementation deemed it beneficial. Given that Borkenhagen provides both the motivation and the structure to add in additional signals, Examiner believes it would be an obvious distinction to use a PCI signal to switch threads.

26. As per Claim 18, Borkenhagen teaches: The method of claim 15. but fails to teach:

wherein the user-specified parameter specifies "sram Swap", and performing a swapping comprises swapping out the current context corresponding to the first thread and waking up the swapped, current context when an SRAM signal associated with the first thread is received.

While Borkenhagen does not specifically teach that one of the parameters is a "sram swap", dealing with the SRAM signal, Borkenhagen does teach a large number of signals that can trigger a context switch, as shown in Columns 14 and 15, and also notes that there are a large number of extra bits available for future use. Examiner is taking Official Notice that one of ordinary skill in the art would have been motivated to

make use of any number of other signals, including the SRAM signal, to trigger a context switch, if their particular implementation deemed it beneficial. Given that Borkenhagen provides both the motivation and the structure to add in additional signals, Examiner believes it would be an obvious distinction to use the SRAM signal to switch threads.

27. As per Claim 19, Borkenhagen teaches: The method of claim 15, but fails to teach:

wherein the user-specified parameter specifies "sdram Swap", and performing a swapping comprises swapping the current context corresponding to the first thread and waking up the swapped, current context when an SDRAM signal associated with the first thread is received.

While Borkenhagen does not specifically teach that one of the parameters is a "sdram swap", dealing with the SDRAM signal, Borkenhagen does teach a large number of signals that can trigger a context switch, as shown in Columns 14 and 15, and also notes that there are a large number of extra bits available for future use. Examiner is taking Official Notice that one of ordinary skill in the art would have been motivated to make use of any number of other signals, including the SDRAM signal, to trigger a context switch, if their particular implementation deemed it beneficial. Given that Borkenhagen provides both the motivation and the structure to add in additional signals, Examiner believes it would be an obvious distinction to use the SDRAM signal

to switch threads.

28. As per Claim 20, Borkenhagen teaches: The method of claim 15, but fails to teach:

wherein the user-specified parameter specifies "inter_thread" which swaps out the current context corresponding to the first thread and wakes up the swapped, current context when an interthread signal associated with the first thread is received.

While Borkenhagen does not specifically teach that one of the parameters is an "inter_thread" signal, dealing with the interthread signal, Borkenhagen does teach a large number of signals that can trigger a context switch, as shown in Columns 14 and 15, and also notes that there are a large number of extra bits available for future use. Examiner is taking Official Notice that one of ordinary skill in the art would have been motivated to make use of any number of other signals, including the interthread signal, to trigger a context switch, if their particular implementation deemed it beneficial. Given that Borkenhagen provides both the motivation and the structure to add in additional signals, Examiner believes it would be an obvious distinction to use the interthread signal to switch threads.

Response to Arguments

29. Applicant's arguments, with respect to the rejection(s) of the claims have been fully considered and are persuasive, in that Borkenhagen does not explicitly teach a plurality of microengines, each of which containing a context arbiter of the claimed

functionality. Therefore, the rejection has been withdrawn. However, upon further consideration, a new ground(s) of rejection is made in view of Borkenhagen.

30. Examiner notes that a further defining of either the micro engines, the processor, or both, may help to overcome the current rejection, if they are in fact different than the Examiners current interpretation. However, given the current, broad definition of the micro engine (in that there are a plurality of them, and that each has a context arbiter), Examiner believes the current rejection appropriately reads upon the claims (where each processor of a multi-core processor reads upon the microengines), but a further narrowing of the microengines may make such an interpretation impossible. If Applicant has any questions about the rejection, or would like the Examiners opinion on whether or not any suggested claim language would overcome the current rejection, Applicant is welcome to contact the Examiner to arrange a time for an interview.

Conclusion

31. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL.** See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not

mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Robert Fennema whose telephone number is (571)272-2748. The examiner can normally be reached on Monday-Thursday, 9:30-6:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Eddie P Chan/
Supervisory Patent Examiner, Art Unit 2183

Robert Fennema
Examiner
Art Unit 2183

RF